Claims

What is claimed is:

1. An apparatus for modeling at least one aspect of a software artifact, said apparatus comprising an arrangement for providing extension types, each extension type comprising an ordered tuple of a plurality of element types, each of the element types corresponding to different class hierarchies.

2. The apparatus according to Claim 1, wherein each extension type comprises an extension or variation of element types.

3. The apparatus according to Claim 1, wherein said extension types are adapted to compose classes horizontally.

4. The apparatus according to Claim 1, wherein each extension type is adapted to masquerade as any associated element type.

5. The apparatus according to Claim 1, wherein each extension type is a subtype of its associated element types.

6. The apparatus according to Claim 1, wherein:

each extension type has a size corresponding to the number of elements associated with the extension type; and

given two extension types α and β , a sub-type relation $\alpha <: \beta$ is definable as follows:

$|\alpha| >= |\beta|$; and

$\alpha(0) <: \beta(0), \alpha(1) <: \beta(1), \ldots \alpha(|\beta|-1) <: \beta(|\beta|-1)$.

7. The apparatus according to Claim 1, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta <: \alpha$:

a method dispatch *p.m* comprises starting at the element type $\beta(0)$ and walking up the class hierarchy of $\beta(0)$ to find the closest *m,* wherein if *m* is not defined in the class hierarchy of $\beta(0)$, then *m* is sought in the $\beta(1)$ class hierarchy and, if needed, in one or more iteratively successive class hierarchies, until found.

8. The apparatus according to Claim 1, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta <: \alpha$:

a method dispatch $p*m$ comprises, for each element type $\beta(i)$, in the order $i=0, \ldots, |\beta|-1$, walking up the class hierarchy of $\beta(i)$ to find the closest m in $\beta(i)$ and dispatching the method m (if found), whereby a type error arises if m is not defined in at least one of the class hierarchies $\beta(i)$, $i=0, \ldots, |\beta|-1$.

9. The apparatus according to Claim 1, wherein, with α being the extension type of a variable p and β being the runtime extension type of the object pointed by p, so that $\beta<:\alpha$:

a method dispatch $p(1,3,4).m$ comprises reviewing only a class hierarchy of $\beta(1)$, $\beta(3)$, and $\beta(4)$to find the closest m, wherein a type error arises if m is not defined in any of $\beta(1)$, $\beta(3)$, or $\beta(4)$.

10. The apparatus according to Claim 1, wherein, with α being the extension type of a variable p and β being the runtime extension type of the object pointed by p, so that $\beta<:\alpha$:

a method dispatch $p(1,3,4)*m$ comprises reviewing only a class hierarchy of $\beta(1)$, $\beta(3)$, and $\beta(4)$to find the closest m in $\beta(i)$ and dispatching the method m if found, whereby a type error arises if in any of the class hierarchies to which $\beta(1)$, $\beta(3)$, or $\beta(4)$ belongs m is not defined.

11. A method of modeling at least one aspect of a software artifact, said method comprising the step of providing extension types, each extension type comprising an ordered tuple of a plurality of element types, each of the element types corresponding to different class hierarchies.

12. The method according to Claim 11, wherein each extension type comprises an extension or variation of element types.

13. The method according to Claim 11, wherein the extension types are adapted to compose classes horizontally.

14. The method according to Claim 11, wherein each extension type is adapted to masquerade as any associated element type.

15. The method according to Claim 11, wherein each extension type is a subtype of its associated element types.

16. The method according to Claim 11, wherein:

each extension type has a size corresponding to the number of elements associated with the extension type; and

given two extension types α and β , a sub-type relation $\alpha <: \beta$ is definable as follows:

$|\alpha| >= |\beta|$; and

$\alpha(0) <: \beta(0), \alpha(1) <: \beta(1), \ldots \alpha(|\beta|-1) <: \beta(|\beta|-1)$.

17. The method according to Claim 11, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta <: \alpha$:

a method dispatch *p.m* comprises starting at the element type $\beta(0)$ and walking up the class hierarchy of $\beta(0)$ to find the closest *m,* wherein if *m* is not defined in the class hierarchy of $\beta(0)$, then *m* is sought in the $\beta(1)$ class hierarchy and, if needed, in one or more iteratively successive class hierarchies, until found.

18. The method according to Claim 11, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta <: \alpha$:

a method dispatch *p*m* comprises, for each element type $\beta(i)$, in the order *i=0, ...,* $|\beta|-1$, walking up the class hierarchy of $\beta(i)$ to find the closest *m* in $\beta(i)$ and dispatching

the method *m* (if found), whereby a type error arises if *m* is not defined in at least one of the class hierarchies $\beta(i)$, $i=0, \ldots, |\beta|-1$.

19. The method according to Claim 11, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta<:\alpha$:

a method dispatch *p(1,3,4).m* comprises reviewing only a class hierarchy of $\beta(1)$, $\beta(3)$, and $\beta(4)$to find the closest *m,* wherein a type error arises if *m* is not defined in any of $\beta(1)$, $\beta(3)$, or $\beta(4)$.

20. The method according to Claim 11, wherein, with α being the extension type of a variable *p* and β being the runtime extension type of the object pointed by *p*, so that $\beta<:\alpha$:

a method dispatch *p(1,3,4)*m* comprises reviewing only a class hierarchy of $\beta(1)$, $\beta(3)$, and $\beta(4)$to find the closest *m* in $\beta(i)$ and dispatching the method *m* if found, whereby a type error arises if in any of the class hierarchies to which $\beta(1)$, $\beta(3)$, or $\beta(4)$ belongs *m* is not defined.

21. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for modeling

at least one aspect of a software artifact, said method comprising the step of providing extension types, each extension type comprising an ordered tuple of a plurality of element types, each of the element types corresponding to different class hierarchies.